Blockbusters and Wallflowers: Speeding up Diverse and Accurate Recommendations with Random Walks

Christoffel, Fabian ; Paudel, Bibek ; Newell, Chris ; Bernstein, Abraham

Abstract: User satisfaction is often dependent on providing accurate and diverse recommendations. In this paper, we explore algorithms that exploit random walks as a sampling technique to obtain diverse recommendations without compromising on efficiency and accuracy. Specifically, we present a novel graph vertex ranking recommendation algorithm called RP3 that re-ranks items based on 3-hop random walk transition probabilities. We show empirically, that RP3 provides accu- rate recommendations with high long-tail item frequency at the top of the recommendation list. We also present approx- imate versions of RP3 and the two most accurate previously published vertex ranking algorithms based on random walk transition probabilities and show that these approximations converge with increasing number of samples.

DOI: https://doi.org/10.1145/2792838.2800180

Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks

Fabian Christoffel[a], Bibek Paudel[a], Chris Newell[b], Abraham Bernstein[a]
[a]Department of Informatics, University of Zurich, Zurich, Switzerland
[b]Research and Development, British Broadcasting Corporation, London, United Kingdom
fabian.christoffel@uzh.ch, {paudel, bernstein}@ifi.uzh.ch, chris.newell@bbc.co.uk

ABSTRACT

User satisfaction is often dependent on providing accurate and diverse recommendations. In this paper, we explore scalable algorithms that exploit random walks as a sampling technique to obtain diverse recommendations without compromising on accuracy. Specifically, we present a novel graph vertex ranking recommendation algorithm called RP^3_β that re-ranks items based on 3-hop random walk transition probabilities. We show empirically, that RP^3_β provides accurate recommendations with high long-tail item frequency at the top of the recommendation list. We also present scalable approximate versions of RP^3_β and the two most accurate previously published vertex ranking algorithms based on random walk transition probabilities and show that these approximations converge with increasing number of samples.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Information Filtering

Keywords: top-N recommendation; item ranking; diversity; long-tail; bipartite graph; random walks; sampling

1. INTRODUCTION

Users increasingly rely on recommender systems to choose movies, books, restaurants and other items. These systems are usually based on the assumption that users prefer items similar to those they previously liked or those liked by other users with similar preferences. However, this approach has some deficiences. Pariser [18] introduced the term "filter bubble" to describe how personalized recommendations can isolate people from diverse viewpoints or products. This has also led to the concern that recommender systems may reinforce the blockbuster nature of media [8] due to their promotion of already popular products. Also, the focus on the predictive accuracy of recommender systems can lead to a bias towards popular items over more specialized items. In other words, systems that are optimized for accuracy tend to produce unsurprising and boring recommendations.

User satisfaction depends on many factors such as variety, new experiences and serendipitous discovery which are not captured by accuracy metrics. These factors depend on finding suitable long tail items, which raise user satisfaction and, in turn, profitability [11]. Hence, a recent trend is to build recommender systems that do not only focus on optimizing the accuracy but also consider the diversity of recommendations [1, 2, 23, 24]. However, these can be conflicting goals as increasing diversity may produce irrelevant recommendations.

Recently, it was shown that approximations based on random graph walks can be used for accurate recommendations [6]. These algorithms are more accurate than previously presented vertex ranking methods (e.g., [9]) with the additional benefit of being computationally efficient and scalable. In this paper, we *explore scalable algorithms that exploit random walks as a sampling technique to obtain diverse recommendations without compromising on accuracy.*

Specifically, our contributions are: First, we introduce RP^3_β, a simple item popularity dependent re-ranking procedure of P^3 [6]. We show using three implicit feedback datasets (two public, one enterprise) that RP^3_β augments long-tail item recommendations while keeping accuracy high. Second, we empirically compare the performance of vertex ranking algorithms [9, 6, 23] including our own RP^3_β with traditional state-of-the-art methods. We find that some vertex ranking algorithms achieve comparable or better performance than the traditional ones. Third, we present scalable approximation algorithms for RP^3_β, P^3_α [6], as well as H_λ [23] based on random walk sampling. In a detailed evaluation we show that these methods converge to the performance scores of exact calculations. Last, we analyze the trade-off between sampling size (i.e., number of performed random walks) versus accuracy and diversity performance and find that RP^3_β provides a useful trade-off between accuracy, diversity, and sample size.

The remainder of this paper begins with a description of our data model and notations. We present a literature review in Section 3. We then describe our method RP^3_β and our approximations for P^3_α, RP^3_β, and H_λ. The experimental results are presented in Section 6 followed by conclusions.

2. MODEL

The recommendation algorithms studied in this paper try to rank the items in the training set for each user in the test set by decreasing appreciation. The algorithms are based on walks over the graph $G = (V, E)$ constructed from the users' feedback on items (*user-item-feedback graph*). The vertices V of G represent the union of the two entity sets users U and items I (i.e., $V = U \cup I$) in the training data.

RecSys '15, September 16-20, Vienna, Austria
ACM 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2800180 .

If user $u \in U$ implicitly rated item $i \in I$ in the training phase (e.g., by accessing the item) then the graph's edge set $E \subseteq U \times I$ contains the edge $e = \{u, i\}$. As E contains no other edges, G is bipartite. All edges in the graph are unweighted/undirected and no parallel edges exist. Edge weights or parallel edges (e.g., based on rating values or the number of interactions) could be used for a more accurate representation of the users preference profile, but we do not consider this extension in the presented work.

The square matrix $A \in \{0,1\}^{|V|\times|V|}$ is the *adjacency matrix* of G. Since edges of G are undirected, A is symmetric. The entry a_{ij} of A is 1 for two connected vertices i and j, and 0 otherwise. $D^{|V|\times|V|}$ is the diagonal *degree matrix* of G with $d_{ii} = \sum_{j=1}^{|V|} a_{ij}$. Assuming all diagonal elements of D are non-zero (i.e., no unconnected vertices), its inverse D^{-1} is given by (d_{ii}^{-1}), and hence cheap to compute.

A *random walk* process on G can be seen as a discrete Markov chain, where a walker starts on a vertex $v(0)$ and at each time step moves to one of its neighbors chosen randomly. After s steps, the sequence of vertices visited by the walker $\langle v(0), v(1), \dots v(s)\rangle$ forms a Markov chain. The probability of transitioning from a vertex i to j is $p_{ij} = a_{ij}/d_{ii}$. Hence, the corresponding transition matrix $P^{|V|\times|V|}$ for one-step ($s = 1$) random walks is given by $P = D^{-1}A$. Furthermore, we obtain the s-step random walk transition probability matrix (we refer to its elements with p_{ij}^s) with

$$P^s = (D^{-1}A)^s. \tag{1}$$

Since we want to rank items i for users u, this paper considers random walks starting at user vertices and ending at item vertices (i.e., having an odd number of steps). To denote transition probabilities estimated using random walk samples (see Section 5), we write $\hat{p}_{ij}^s$. In general, an estimate of a random variable X is represented as $\hat{X}$.

3. RELATED WORK

In this section we discuss previous work on vertex ranking algorithms, sampling techniques, and diversity.

Graph based algorithms: The use of a graph-based model for recommendations was first introduced in [3]. To apply a bipartite user-item-feedback graph G was proposed in [14] and several projects [4, 5, 6, 9, 12, 15, 16, 22] extended this approach. We classify them as *vertex ranking algorithms* because their main idea is to rank the vertices in the graph based on their similarities with the target user and use the ranking to generate recommendations. Fouss *et al.* [9] introduced the idea of using random walks on G to rank the vertices. Vertices are ranked or scored based on quantities like hitting time, average commute time or the entries in the Moore-Penrose pseudo inverse of the Laplacian matrix of the graph (L^+). ItemRank [12] also scores vertices based on random walks on the graph, but uses a graph representing item correlations.

Random walk approximations: Cooper *et al.* [6] proposed three new methods called P^3, P^5, and P_α^3 based on random walks on G. They rank vertices based on transition probabilities after short random walks between users and items. P^3 and P^5 perform random walks of fixed length 3 and 5, respectively, starting from a target user vertex. P_α^3, which raises the transition probabilities to the power of α, is more accurate than the methods proposed in [9] and [12]. They also show that approximating the P^3 and P^5 rankings with time- and memory-efficient random walk sampling is more scalable compared to methods based on matrix calculations, i.e., the methods can be applied to larger datasets.

Diversity in recommendations: The erstwhile focus of recommender systems research on improving accuracy (how well the system can predict future user behavior) was criticized as being detrimental to the goals of improving user experience and sales diversity [7, 17]. A recent trend, therefore, is to focus on the diversity of recommendations along with accuracy. Notions of novelty and diversity in recommender systems, as well as measures to quantify, and methods to improve them have been described by various authors [1, 2, 13, 21, 24]. Optimizing only for diversity will cause highly varied but irrelevant recommendations. Therefore it is necessary to find diverse recommendations that are also accurate. Zhou *et. al* [23] use vertex ranking algorithms to improve diversity and accuracy. Specifically, they describe a hybrid method (Hybrid or H_λ) that combines the ranking of an accurate with the ranking of a diverse algorithm.

In this work, we focus on the task of generating diverse and accurate recommendations with vertex ranking algorithms using scalable random walk sampling. To the best of our knowledge, this is the first work to bring together the three streams reviewed above: graph-based approaches, random walk approximation, and diversity. There are different notions of diversity in recommendation lists. Following [2, 23], we use three top-k measures to evaluate recommendation quality in terms of diversity: personalization, item-space coverage, and surprisal. Surprisal assures inclusion of long-tail items at the top of recommendation list, item-space coverage assures that varying long-tail items are considered, and personalization measures how much the recommendation list differs between users. We introduce RP_β^3, a novel algorithm to optimize the accuracy and diversity trade-off by re-ranking the P^3 item ranking. RP_β^3 benefits from the good scalability of approximating P^3 with random walk sampling. Also, we present approximations for H_λ and P_α^3 with the same sampling approach.

4. RP_β^3: POPULARITY-BASED RE-RANKING

In our experiments, we observed (see Section 6.3) that the ranking of items according to the transition probability matrix P^3 is strongly influenced by the popularity (i.e., vertex degree) of items. Hence, for most users the well known blockbuster (or high-degree) items dominate the recommendation lists. To compensate for the influence of popularity and to leverage recommendation of items from the long-tail, we introduce a simple re-ranking procedure dependent on item-popularity. The original score of item i for user u given by p_{ui}^3 (the transition probability after a random walk of length three from u to i). We re-weight the score with

$$\tilde{p}_{ui}^3 = \frac{p_{ui}^3}{d_{ii}^\beta}\ , \text{ where } \beta \in \mathbb{R} \text{ and } \beta > 0.0. \tag{2}$$

For two items i, j ($i \neq j$), a user u with $p_{ui}^3 = p_{uj}^3$ (equal probability of reaching the items from the user in a three-step random walk), and $d_{ii} < d_{jj}$ (i has a lower degree), the effect of our re-weighting is that i is ranked higher than j ($\tilde{p}_{uj}^3 < \tilde{p}_{ui}^3$). These items would have received equal scores without the re-weighting. We refer to this recommendation algorithm as RP_β^3. When we set the parameter $\beta = 0.0$, then RP_β^3 produces the same score as P^3 as $d_{ii}^{\beta=0} = 1$.

5. APPROXIMATING P^3_α, RP^3_β, AND H_λ

In a recent paper, Cooper *et al.* [6] compare two approaches to calculate vertex transition probabilities: by exact calculations using matrix algebra and by approximation via random walk sampling. It is shown that the latter approach is time- and memory-efficient, allowing the application on larger datasets with only limited impact on accuracy. However, they do not describe a sampling procedure for their algorithm P^3_α. Similarly, H_λ, a vertex-ranking algorithm that increases both recommendation accuracy and diversity [23], could also be made more scalable with a sampling procedure instead of exact calculations with matrix algebra.

This section introduces a novel random walk sampling procedure for both of these two algorithms as well as our reranking algorithm RP^3_β .

5.1 Sampling as a Bernoulli Process

In order to estimate transition probabilities for user u using samples, we start multiple s-step random walks from u. We store the number of times each item i is visited by walks at the s^{th} step. For reasons of efficiency, we would like to estimate the probabilities only based on these counts and the degrees of vertices traversed by the path. This sampling procedure can be modeled as a Bernoulli process as follows:

Denote the path traversed by the r^{th} random walk of length s starting at u as $\pi_u^{r,s}$. Then define $I_r^s(u,i) = c_{rw}(\pi_u^{r,s})$ if i is the s^{th} vertex in path $\pi_u^{r,s}$, i.e., if $\pi_u^{r,s}[s] = i$, and $I_r^s(u,i) = 0$ otherwise. The quantity $c_{rw}(\pi_u^{r,s})$ is a function of the vertices' degrees in the path (and varies for different algorithms). For simplicity, we use $I_r(u,i)$ for random walks of a fixed given length (e.g., $s \in \{3,5\}$). Next, define $\tau(u,i)$ as the score of item i for user u and $\hat{\tau}(u,i)$ as its estimator. When sampling N random walks starting from u, the estimator can be defined as $\hat{\tau}(u,i) = \frac{1}{N}\sum_{r=1}^{N} I_r(u,i)$. Given the law of large numbers, the expected value for $\hat{\tau}(u,i)$ is $E[\hat{\tau}(u,i)] = \tau(u,i)$. Also, walks are independent and $I_r \in [0,\psi]$ is i.i.d, where ψ is the maximum possible value for c_{rw}.

Similar to [20], we can use Hoeffding's inequality to show that the rate of convergence is exponential. Furthermore, using Union bound, the probability of the ϵ-approximate estimate for *any* user being less than δ is given as:

$$P(\exists u \in U, |\hat{\tau}(u,i) - \tau(u,i)| \geq \epsilon) \;\leq\; 2|U|\; exp(-\tfrac{2N\epsilon^2}{\psi^2}) \;\leq\; \delta$$

This provides a lower bound for N as $\frac{\psi^2}{2\epsilon^2} log \frac{2|U|}{\delta}$. For a fixed ϵ and δ, the number of walks required increases with ψ, which depends on the algorithm in use and degree distribution of the graph (due to different forms of c_{rw}).

For our method RP^3_β (Section 4), $c_{rw}(\pi_u^{r,s})$ is simply $1/d_{ii}^\beta$, hence, $\psi = 1/\underset{i}{\mathrm{argmin}}(d_{ii}^\beta)$ and the scores can be estimated as described above. For P^3_α and H_λ, $c_{rw}(\pi_u^{r,s})$ takes more complicated forms, which we discuss below. Hereafter, we denote a path simply as π.

5.2 Approximating P^3_α and RP^3_β

Ordering items in descending order according to the transition probabilities of random walks of length three (P^3, $s = 3$) is an accurate recommendation strategy, named P^3 in [6] and ProbS in [23]. The accuracy of this algorithm can be further improved by raising each entry of the transition probability matrix P^1 ($s = 1$) to the power of a parameter $\alpha \in \mathbb{R}$ resulting in an algorithm called P^3_α by [6]. It follows from (1) that entries of the matrix P^1 raised to the power of α are calculated as $p^1_{ui_\alpha} = (p^1_{ui})^\alpha = (a_{ui}/d_{uu})^\alpha$, where $a_{ui} \in A$ (entry in adjacency matrix) and $d_{uu} \in D$ (entry in degree matrix). The transition probability $p^3_{ui_\alpha} \in P^3_\alpha$ from user u to item i after a random walk of length three is obtained by:

$$p^3_{ui_\alpha} = \sum_{v=1}^{|V|}\sum_{j=1}^{|V|} p_{uj_\alpha} p_{jv_\alpha} p_{vi_\alpha} = \sum_{v=1}^{|V|}\sum_{j=1}^{|V|} \left(\frac{a_{uj}}{d_{uu}}\right)^\alpha \left(\frac{a_{jv}}{d_{jj}}\right)^\alpha \left(\frac{a_{vi}}{d_{vv}}\right)^\alpha \tag{3}$$

Since the graph G defined in Section 2 is both bipartite (there are no edges from users to users or from items to items) and all entries in the adjacency matrix A are either 0 or 1, we can simplify (3) as:

$$p^3_{ui_\alpha} = \sum_{v=1}^{|I|}\sum_{j=1}^{|U|} \frac{a_{uj}a_{jv}a_{vi}}{(d_{uu}d_{jj}d_{vv})^\alpha} \tag{4}$$

The term $a_{uj}a_{jv}a_{vi}$ in (4) is 1 if a path of length three starting from user u, through item j and user v, to item i exists in the graph G and is 0 otherwise. Hence, $p^3_{ui_\alpha}$ is the aggregate of all paths of length three between user u and item i, where each path $\pi = \langle U_u, I_j, U_v, I_i\rangle$ contributes $c_\pi^{\mathrm{P}^3_\alpha} = \frac{1}{(d_{uu}d_{jj}d_{vv})^\alpha}$ to the total transition probability from user u to item i.

When approximating (4) with random walk sampling, one needs to take into account that some walks are more likely to be followed randomly than others. The probability of following the path from u via the item j and user v to item i in a random walk is dependent on three decisions. First, at user u, one needs to follow the edge that connects u to item j. The probability of randomly picking this edge is equal to the inverse of the degree of u: $\Pr(u \to j) = \frac{1}{d_{uu}}$. Next, the same procedure needs to be repeated at j and v, resulting in $\Pr(j \to v) = \frac{1}{d_{jj}}$ and $\Pr(v \to i) = \frac{1}{d_{vv}}$. Given that these three "choices" are independent, the probability $Pr(\pi)$ that one follows the path π is equal to

$$\Pr(\pi) = \Pr(u \to j)\Pr(j \to v)\Pr(v \to i) = \frac{1}{d_{uu}d_{jj}d_{vv}}. \tag{5}$$

Hence, when approximating with random walks, we are more likely to follow paths traversing vertices of low degrees than to follow paths traversing vertices of high degrees. Since an exact calculation of (4) requires following each path exactly once, random walk sampling needs to discount the contribution of paths with high probabilities (as we may by chance follow them many times), and boost the contribution of paths with low probabilities (as we may by chance follow them only few times). Consequently, to approximate the transition probability $\hat{p}^3_{ui_\alpha}$, we weigh a path contribution $c_\pi^{\mathrm{P}^3_\alpha}$ with the inverse of its occurrence probability $(\Pr(\pi)^{-1})$ resulting in an overall weight $c_{rw}^{\hat{\mathrm{P}}^3_\alpha}$ for a random walk:

$$\begin{aligned} c_{rw}^{\hat{\mathrm{P}}^3_\alpha} &= c_\pi^{\mathrm{P}^3_\alpha} * \Pr(\pi)^{-1} \\ &= \underbrace{\frac{1}{(d_{uu}d_{jj}d_{vv})^\alpha}}_{\substack{\text{path contribution}}} * \underbrace{d_{uu}d_{jj}d_{vv}}_{\substack{\text{inverted path probability}}} = \underbrace{(d_{uu}d_{jj}d_{vv})^{1-\alpha}}_{\substack{\text{random walk contribution}}} \end{aligned} \tag{6}$$

We can simplify $c_{rw}^{\hat{\mathrm{P}}^3_\alpha}$ to $(d_{jj}d_{vv})^{1-\alpha}$ since d_{uu} takes the same value for all random walks of the target user u and, hence, does not influence the item ranking order.

Algorithm 1 Estimating item scores of $\hat{\mathrm{P}}^3_\alpha$, $\mathrm{R}\hat{\mathrm{P}}^3_\beta$, or $\hat{\mathrm{H}}_\lambda$ with random walk sampling.

Require: v_u is the vertex representing user u

```
1: function EstimateItemScores(v_u)
2:     m ← an associative array with default value 0
3:     while !Converged(m) do
4:         v_c ← GetRandomNeighbor(v_u)
5:         d_jj ← GetDegree(v_c)
6:         v_c ← GetRandomNeighbor(v_c)
7:         d_vv ← GetDegree(v_c)
8:         v_c ← GetRandomNeighbor(v_c)
9:         d_ii ← GetDegree(v_c)
10:        m[v_c] ← m[v_c] + c_rw
11:    end while
12:    return m
13: end function
```

Algorithm 1 shows the general principle of how to implement a random walk sampling approximation procedure. With this algorithm we obtain $\hat{\mathrm{P}}^3_\alpha$ item scores by assigning $c_{rw}^{\hat{\mathrm{P}}^3_\alpha}$ to the random walk contribution c_{rw} in line 10. Note that for $\alpha = 1$, the random walk contribution is $(d_{jj}d_{vv})^0$ and degenerates to 1. Hence, the sampling procedure $\hat{\mathrm{P}}^3$ (same as $\hat{\mathrm{P}}^3_{\alpha=1}$) is computationally less demanding, since updating the score of the destination item i of a random walk consists only of incrementing the count of i by one.

To estimate the item ranking of RP^3_β with random walk sampling, we can either first obtain $\hat{\mathrm{P}}^3$ item scores and apply the re-ranking described in Section 4, or replace the random walk contribution $c_{rw}^{\hat{\mathrm{P}}^3} = 1$ by $c_{rw}^{\mathrm{R}\hat{\mathrm{P}}^3_\beta} = 1/d_{ii}^\beta$ and omit the re-ranking. Hence, Algorithm 1 also fully describes $\mathrm{R}\hat{\mathrm{P}}^3_\beta$.

5.3 Approximation of H_λ

Zhou *et al.* [23] define H_λ as a scoring procedure of items using a weighted linear aggregation of scores from two algorithms: HeatS, which is analogous to heat diffusion across the user-item graph and ProbS, which is the same as P^3. W^{H+P} with dimension $|I| \times |I|$ is the transition matrix for H_λ and $f^u \in \{0,1\}^{|I|}$ is the preference profile of target user u, where f_i^u, the i^{th} entry of f^u, is equal to the corresponding entry a_{iu} in the adjacency matrix A. Then, the item scores for user u are calculated as $\widetilde{f}^u = W^{H+P} f^u$. A single entry of W^{H+P} is calculated according to

$$w_{ij}^{H+P} = \frac{1}{d_{ii}^{1-\lambda} d_{jj}^{\lambda}} \sum_{v=1}^{|U|} \frac{a_{iv} a_{jv}}{d_{vv}} \tag{7}$$

where $\lambda \in [0,1]$ is the hybridization parameter for the two basic methods. If we set $\lambda = 0$ or $\lambda = 1$, the ranking of H_λ is equal to the ranking of HeatS or ProbS, respectively. Furthermore, d_{ii} denotes the degree of item i and d_{vv} the degree of user v. The score of item i for the target user u can also be determined according to:

$$\widetilde{f}_i^u = \sum_{j=1}^{|I|} a_{ju} \frac{1}{d_{ii}^{1-\lambda} d_{jj}^{\lambda}} \sum_{v=1}^{|U|} \frac{a_{jv} a_{iv}}{d_{vv}} = \sum_{j=1}^{|I|} \sum_{v=1}^{|U|} \frac{a_{ju} a_{jv} a_{iv}}{d_{ii}^{1-\lambda} d_{jj}^{\lambda} d_{vv}} \tag{8}$$

We can apply the same rationale for the deduction of a random walk simulation algorithm of H_λ as used for P^3_α: the term $a_{ju}a_{jv}a_{iv}$ in (8) is 1 if a path of length three from user u to item i exists in the graph G and 0 otherwise. Hence, $\widetilde{f}_i^u$ is the aggregate of all paths of length three between user u and item i, where a single path contributes $c_\pi^{\mathrm{H}_\lambda} = \frac{1}{d_{ii}^{1-\lambda} d_{jj}^{\lambda} d_{vv}}$ to the score of item i for user u. Because (8) (similar to (4) for P^3_α) requires that each path contribution $c_\pi^{\mathrm{H}_\lambda}$ is counted once, we need to weight $c_\pi^{\mathrm{H}_\lambda}$ by the inverted path probability $\Pr(\pi)^{-1}$. The random walk path contribution $c_{rw}^{\hat{\mathrm{H}}_\lambda}$ for the random walk sampling approximation algorithm ($\hat{\mathrm{H}}_\lambda$) is calculated according to:

$$c_{rw}^{\hat{\mathrm{H}}_\lambda} = c_\pi^{\mathrm{H}_\lambda} * \Pr(\pi)^{-1} = \frac{d_{uu} d_{jj} d_{vv}}{d_{ii}^{1-\lambda} d_{jj}^{\lambda} d_{vv}} = \frac{d_{uu} d_{jj}^{1-\lambda}}{d_{ii}^{1-\lambda}} \tag{9}$$

Again, we can further simplify $c_{rw}^{\hat{\mathrm{H}}_\lambda}$ to $\frac{d_{jj}^{1-\lambda}}{d_{ii}^{1-\lambda}}$, since d_{uu} is the same value for all random walks for the target user u, and hence does not influence the item ranking order. With Algorithm 1 we obtain $\hat{\mathrm{H}}_\lambda$ item scores by assigning $c_{rw}^{\hat{\mathrm{H}}_\lambda}$ to c_{rw}.

6. EXPERIMENTS AND EVALUATION

This section provides a succinct introduction to the experimental methodology and then turns to the main questions of the paper: First, it explores if RP^3_β improves accuracy and diversity. Then it explores a general comparison between vertex ranking and traditional algorithms. It closes with a thorough comparison between P^3_α, RP^3_β, and H_λ and our approximate versions $\hat{\mathrm{P}}^3_\alpha$, $\mathrm{R}\hat{\mathrm{P}}^3_\beta$, and $\hat{\mathrm{H}}_\lambda$.

6.1 Methodology

Datasets: We used the MovieLens-M[1], iPlayer, and BookCrossing [24] datasets (see Table 1 for properties). Whilst Movie-Lens-M and BookCrossing are public, the iPlayer training dataset consists of the viewing logs of the BBC VoD system from the week of February 15-21, 2014, and the test data of the following week's logs, where only interactions with a single show longer than 5 minutes were considered. From the log data of the test week, we randomly selected 5'000 users that were also active during the training week. Since this work addresses recommendation generation based on implicit user feedback, we neglected the rating values available in MovieLens-M and BookCrossing for training and testing of the evaluated recommenders.

Set-Up: We extended the Java port of the MyMediaLite [10] recommender system framework[2] with (i) a set of metrics (see the following paragraphs) to measure recommendation performance according to the diversity dimensions

	MovieLens-M	iPlayer	BookCrossing
total ratings	1'000'047	4'703'471	369'195
total users	6'038	655'846	4'052
total items	3'706	808	18'280
Training: # ratings	700'047	4'691'493	258'436
min. / avg. ratings per user	10 / 115.9	1 / 7.2	15 / 63.8
min. / avg. ratings per item	1 / 188.9	5 / 5806.3	1 / 14.1
Sparsity	0.031	0.0089	0.0035
Graph Diameter (approx.*)	6	6	7
Test: # ratings	300'000	14'616	110'759
min. / avg. ratings per user	1 / 49.7	1 / 2.9	1 / 27.3
min. / avg. ratings per item	1 / 85.4	1 / 23.4	1 / 6.6
min. train-ratings for user	10	1	15
min. train-ratings for item	1	18	5

Table 1: Dataset Properties. *`PseudoDiameter` of *Mathematica 10®*.

[1]MovieLens-M: `grouplens.org/datasets/movielens`
[2]Java port: `github.com/jcnewell/MyMediaLiteJava`

	Recommender	AUC	Prec@20	GiniD@20	Pers@20	Surp@20
MovieLens-M	Perfect	1.0000	0.835	0.218	0.927	4.01
	RP^3_β ($\beta = 0.8$)	**0.9287**	0.341	0.172	0.941	**3.79**
	RP^3_β ($\beta = 0.7$)	**0.9260**	**0.359**	0.080	0.862	2.80
	H_λ ($\lambda = 0.1$)	**0.9240**	0.338	0.100	0.913	**3.63**
	H_λ ($\lambda = 0.2$)	0.9214	0.347	0.052	0.831	2.58
	BPRMF (d=50)	0.9211	0.324	0.189	0.952	3.22
	BPRMF (d=200)	0.9197	0.333	0.145	0.932	2.96
	WI-kNN (k=150)	0.9180	**0.353**	0.090	0.918	2.75
	WI-kNN (k=200)	0.9178	**0.354**	0.085	0.912	2.71
	I-kNN (k=150)	0.9138	0.283	**0.221**	**0.973**	3.62
	I-kNN (k=50)	0.9056	0.295	**0.204**	**0.968**	3.47
	P^3_α ($\alpha = 1.8$)	0.9028	0.259	0.027	0.644	2.13
	P^3_α ($\alpha = 1.5$)	0.9011	0.263	0.015	0.565	1.96
	P^3	0.8910	0.252	0.011	0.497	1.88
	L^+	0.8811	0.215	**0.218**	**0.971**	**4.22**
	#3-Paths	0.8672	0.234	0.010	0.449	1.86
	P^5	0.8600	0.217	0.009	0.410	1.84
	MostPop	0.8514	0.210	0.009	0.401	1.84
	Random	0.5018	0.015	0.900	0.994	6.33
iPlayer	Perfect	0.9618	0.120	0.068	0.172	8.21
	H_λ ($\lambda = 0.2$)	**0.8972**	**0.059**	0.251	0.805	4.94
	RP^3_β ($\beta = 0.7$)	**0.8949**	**0.059**	**0.327**	**0.848**	**5.29**
	WI-kNN (k=150)	**0.8911**	**0.058**	0.195	0.734	4.55
	P^3_α ($\alpha = 1.5$)	0.8804	0.051	0.139	0.617	4.13
	P^3	0.8785	0.049	0.108	0.567	3.95
	BPRMF (d=50)	0.8756	0.056	0.211	0.734	4.54
	#3-Paths	0.8630	0.043	0.077	0.490	3.75
	I-kNN (k=50)	0.8560	0.033	**0.340**	**0.867**	**6.11**
	I-kNN (k=10)	0.8056	0.046	**0.309**	**0.869**	**5.89**
	MostPop	0.7506	0.024	0.038	0.163	3.31
	Random	0.4950	0.004	0.954	0.968	8.01
BookCrossing	Perfect	1.0000	0.663	0.266	0.828	7.80
	H_λ ($\lambda = 0.6$)	**0.8291**	**0.080**	0.109	0.854	5.83
	H_λ ($\lambda = 0.5$)	**0.8283**	**0.082**	0.158	0.913	6.42
	RP^3_β ($\beta = 0.3$)	**0.8271**	0.071	0.154	0.876	6.11
	P^3_α ($\alpha = 0.9$)	0.8255	0.059	0.010	0.610	4.44
	P^3	0.8248	0.060	0.015	0.652	4.56
	P^3_α ($\alpha = 1.1$)	0.8235	0.060	0.026	0.703	4.74
	L^+	0.8234	0.033	**0.318**	**0.996**	**9.19**
	P^5	0.8056	0.042	0.002	0.271	4.09
	BPRMF (d=10)	0.7985	0.035	0.100	0.966	6.39
	WI-kNN (k=3200)	0.7825	0.060	0.118	0.955	6.91
	#3-Paths	0.7783	0.048	0.002	0.436	4.14
	BPRMF (d=200)	0.7735	0.048	0.109	0.965	5.87
	I-kNN (k=800)	0.7535	0.048	0.148	**0.976**	**7.38**
	MostPop	0.7180	0.034	0.001	0.111	3.95
	WI-kNN (k=50)	0.6542	**0.083**	**0.236**	0.975	**7.09**
	I-kNN (k=10)	0.5911	0.078	**0.178**	**0.978**	6.97
	Random	0.5010	0.001	0.748	0.999	8.57

Table 2: Accuracy and diversity of all algorithms (ordered by decreasing AUC). Parameterized algorithms are represented by parameter values resulting in maximal AUC and Prec@20 performance. Top 3 numbers per metric highlighted (results from Perfect and Random recommender not considered).

introduced in Section 3 and (ii) a component implementing graph vertex ranking algorithms. Given our focus on implicit feedback we only employed the framework's positive-only feedback components. All computations where executed on a cluster of 16 machines running LINUX with 128 GB RAM and two Intel® Xeon® E5-2680V2 processors (25 MB Cache, 2.80 GHz base frequency, 10 cores, 20 threads).

Accuracy Metrics: We used both the Area Under the ROC curve (AUC) and precision at k (Prec@k). Referring to relevant items (in the test set) as *hits*, AUC is equal to the probability that randomly chosen items are ranked higher than non-hits. Prec@k counts the number of hits among the top-k items of the recommendation list divided by the cut-off level k. Given that users typically only see few recommendations, we chose $k = 20$. Higher values of AUC and Prec@k indicate better accuracy.

Diversity Metrics: We used coverage (Gini-Diversity, GiniD@k), personalization (Pers@k), and surprisal (Surp@k) as diversity metrics and extended the MyMediaLite framework accordingly. Given the already explained rationale, we used $k = 20$. Again, greater values indicate better diversity.

We measure coverage by calculating GiniD@k for the top-k recommendations of all test users [2]. In contrast to the original Gini coefficient, where greater values indicate a more dispersed distribution, GiniD@k increases for a more uniform distribution. GiniD@k is equal to 1 if the frequency in the aggregated recommendation lists is the same for each item, indicating a good coverage.

Pers@k [23] measures the distinctness of the top-k recommendations based on the number of common items averaged over all pairs of generated recommendation sets. A value of Pers@k=1 indicates that none of the items appear more than once among the top-k items of any two recommendation lists, meaning greater personalization.

Surp@k [23] is calculated separately for each recommendation list and averaged over all users. This metric follows the rationale that recommendations of items of low popularity are perceived by the users as unexpected or surprising (unexpectedness given by the self-information of items).

Evaluated Recommendation Algorithms: We compared the performance of our methods with various algorithms proposed in the literature (and listed in Table 2, except for (iii)). These can be divided into the following categories: **(i) Parameter-free vertex ranking algorithms:** #3-Paths (ranks items by the number of paths of length 3 starting at the target user) [6, 14], L^+ (ranks items by the entries in the Moore-Penrose pseudoinverse of the Laplacian matrix) [9], P^3 [6, 23], and P^5 [6]. Due to computational limitations we could not obtain results for P^5 and L^+ for the iPlayer dataset. **(ii) Parameterized vertex ranking algorithms**: P^3_α [6], H_λ [23], and our RP^3_β. **(iii) Approximated/Sampled vertex ranking algorithms** $\hat{P}^3_\alpha$, $R\hat{P}^3_\beta$, and $\hat{H}_\lambda$. **(iv) Other algorithms:** MostPop (global item popularity), Random (random item ranking), weighted (WI-kNN) and unweighted (I-kNN) k-nearest neighbor item-based collaborative filtering using cosine distance as item similarity measure, and BPRMF [19] (a recommender based on a latent factor model obtained with matrix factorization) – all available in MyMediaLite. To facilitate performance comparison, we also calculated the performance of the perfect recommender (Perfect) that places all test items of a user in random order at the top of the recommendation list.

Parameter Tuning: We empirically tune the parameters for parameterized algorithms to maximize the two accuracy metrics. For I-kNN and WI-kNN with MovieLens-M and iPlayer we tested neighborhood sizes k $\in \{10, 50, 100, 150, 200\}$. For the BookCrossing I-kNN was tested for k $\in \{10, 50, 100, 150, 200, 400, 800, 1600\}$ and for WI-kNN we additionally tested k = 3200. Similarly, BPRMF was tested with the latent factors d $\in \{10, 50, 100, 150, 200\}$ for MovieLens-M and BookCrossing. Due to computational limitations, BPRMF could only be tested with d $\in \{10, 50\}$ for iPlayer.[3] For P^3_α, we tested values of $\alpha \in [-0.2, 4.5]$ in steps of 0.1. For RP^3_β, we tested values of $\beta \in [-0.2, 1.2]$ in steps of 0.1. H_λ was tested with values of $\lambda \in [0, 1]$ in steps of 0.1. The best performing parameters with respect to accuracy can be found in parentheses in the results Table 2.

6.2 RP^3_β increases Accuracy and Diversity

The goal of the first set of experiments is to evaluate our re-ranking procedure RP^3_β. To that end we compare it with

[3]Other parameters for BPRMF: 30 stochastic gradient ascent iterations for training, no item bias, iteration length of 5, learning rate α of 0.05, regularization parameter for positive item factors of 0.0025, regularization parameter for negative item factors of 0.00025, and regularization parameter for user factors of 0.0025.

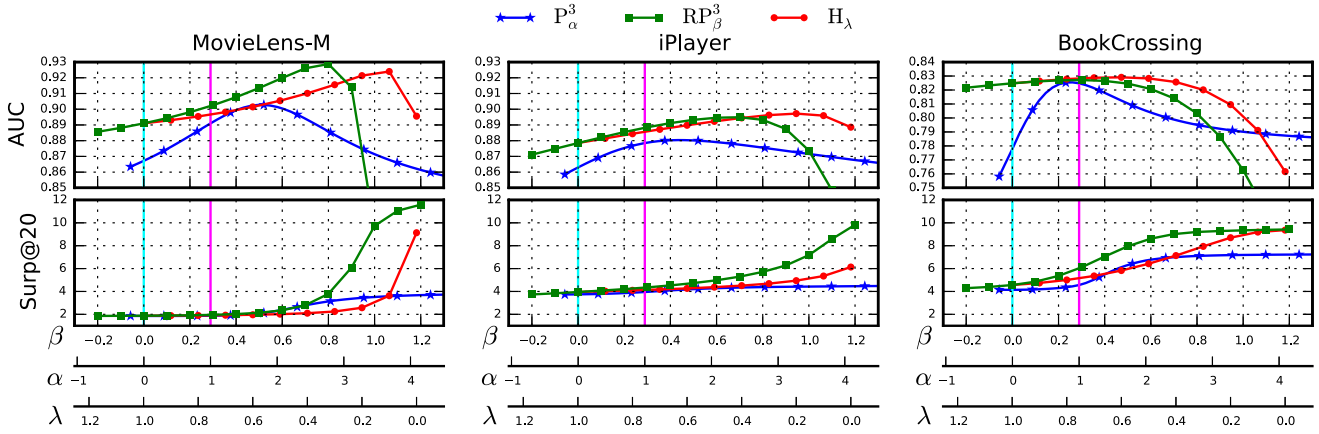

Figure 1: AUC and Surp@20 performance of P^3_α, RP^3_β, and H_λ at different parameter values. The left vertical line (cyan) at $\beta = 0.0$, $\alpha = 0.0$, and $\lambda = 1.0$ indicates the parameter values where RP^3_β and H_λ give the same item ranking as P^3, and P^3_α the ranking of #3-Paths. The right vertical line (magenta) at $\alpha = 1.0$ indicates the parameter value where P^3_α gives the same item ranking as P^3.

the other algorithms evaluated and especially explore its performance compared to P^3_α and H_λ.

As Table 2 shows, the RP^3_β re-ranking increases both accuracy and diversity for all datasets compared to its P^3 basis. Measured by AUC, RP^3_β is the most accurate algorithm for MovieLens-M and second most accurate algorithm after H_λ for iPlayer and BookCrossing. For Prec the results are less favorable: while the performance of RP^3_β is best for MovieLens-M and second best for iPlayer, WI-kNN, I-kNN, and H_λ clearly outperform RP^3_β for BookCrossing. This is possibly due to the lower number of average ratings per item, which may distort our boosting of low degree items.

Cooper *et al.* [6] show that P^3_α improves accuracy over P^3. Our experiments confirm this claim but the accuracy improvements achieved with RP^3_β are even greater than with P^3_α for both AUC and Prec. Furthermore, at parameter values corresponding to maximum accuracy, RP^3_β achieves better GiniD, Pers, and Surp scores than P^3_α. This shows that RP^3_β gives a better trade-off between accuracy and diversity, i.e., at parameter values that achieve highest accuracy it produces more diverse results.

The results do not suggest a winner between RP^3_β and H_λ. In terms of AUC and Prec, RP^3_β has advantage over H_λ for MovieLens-M but not for iPlayer and BookCrossing. For BookCrossing the maximal achieved precision of H_λ is much better than that of RP^3_β. At parameter values corresponding to maximum accuracy, the diversity metric scores for RP^3_β are better for H_λ for MovieLens-M and iPlayer. Again, RP^3_β underperforms compared to H_λ on BookCrossing. Figure 1 graphs the AUC and Surp for P^3_α, RP^3_β, and H_λ for the whole parameter ranges. It shows that the maximally achieved Surp by RP^3_β is better (for MovieLens-M and iPlayer) or comparable (for BookCrossing) to H_λ. The plots for the other accuracy and diversity metrics show similar results as AUC and Surp, respectively, but are omitted due to space considerations. Note that we measured the performance of H_λ only in the originally defined parameter interval ($\lambda \in [0, 1]$). We assume that the diversity performance of H_λ increases further for $\lambda < 0$ at the cost of accuracy.

We can conclude that the new method RP^3_β is a vertex ranking algorithm with top-class accuracy and diversity performance. Tuning of its parameter β allows the trade-off between recommendation accuracy and top-k long-tail item frequency to be controlled.

6.3 Performance of Vertex Ranking Algorithms

In this sub-section we compare the performance of vertex ranking to other recommendation algorithms.

As Table 2 shows, in accordance with [6], P^3 is the most accurate algorithm among the measured parameter-free recommenders (MostPop, P^3, P^5, #3-Paths, and L^+). In particular, P^3 is more accurate than the computationally more expensive L^+ algorithm, which was found to be the most accurate algorithm in an earlier study [9].

For AUC, the parameterized vertex ranking algorithms RP^3_β and H_λ outperform the non-vertex ranking recommendation algorithms I-kNN, WI-kNN, and BPRMF. For Prec, the scores of RP^3_β are high for the MovieLens-M dataset but low for the BookCrossing dataset; the opposite is true for H_λ. WI-kNN, the best performing non-vertex ranking algorithm, performs more consistently and archives comparable results to the best vertex ranking algorithm in terms of Prec.

The parameter free vertex ranking algorithms P^3, P^5, and #3-Paths clearly show lower diversity scores than I-kNN, WI-kNN, and BPRMF in all datasets. This is surprising considering the fact that I-kNN, WI-kNN, and BPRMF are more accurate for some of the datasets (e.g., MovieLens-M). Hence, the better diversity performance of the non-vertex recommenders is not explained by more randomness in their recommendations. Exploring the recommendation lists of P^3, P^5, and #3-Paths reveals that ranking is strongly biased by the item's degree (i.e., favoring blockbusters), resulting in rankings similar to MaxPop. The parameter free L^+ generates diverse recommendations at the cost of low Prec (worse than MostPop for BookCrossing). In terms of AUC it is almost as good as P^3.

Parameterized vertex ranking algorithms provide, besides better accuracy, improved diversity compared to parameter free algorithms. Comparing the diversity performance of the most precise vertex (RP^3_β for MovieLens-M and iPlayer, H_λ for BookCrossing) and non-vertex (WI-kNN for all datasets) ranking recommendation algorithms reveals WI-kNN as the clear winner for BookCrossing: WI-kNN is not only slightly

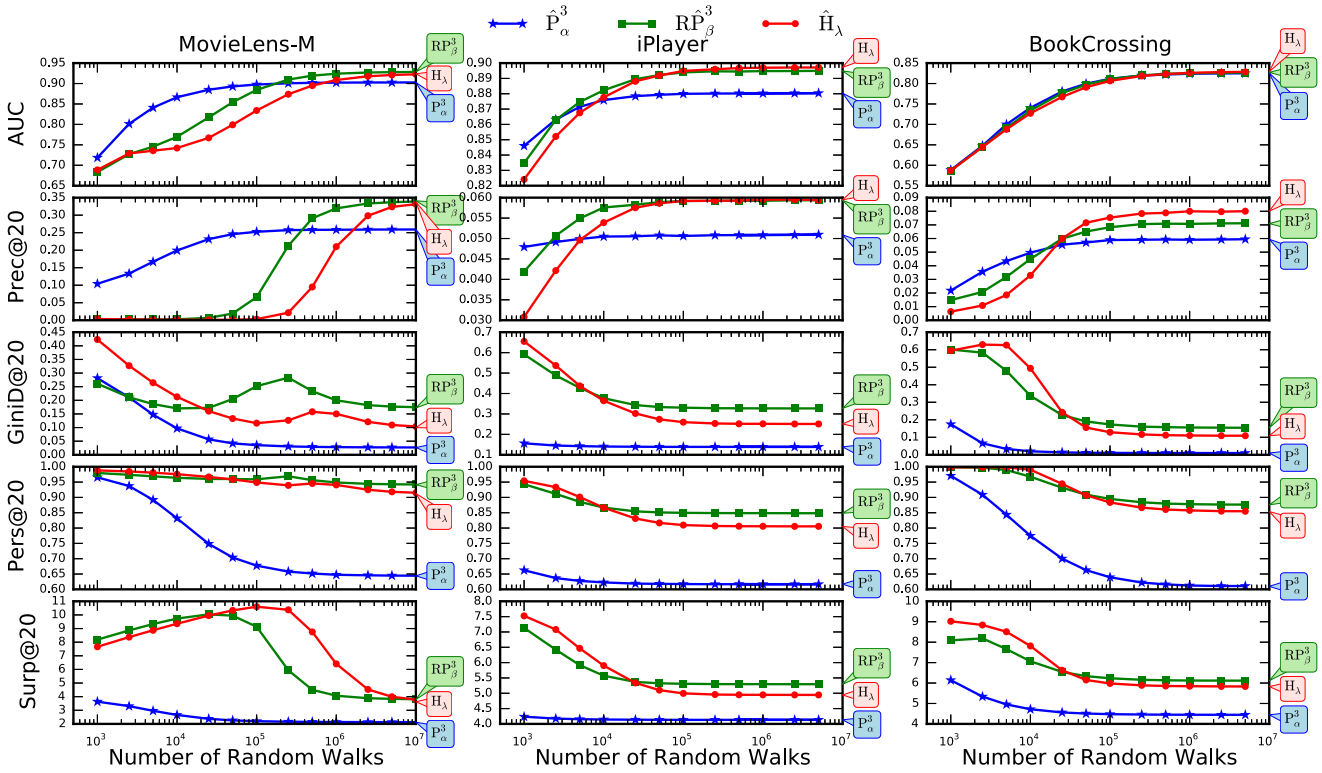

Figure 2: Accuracy and diversity performance of the sampling algorithms $\hat{P}^3_\alpha$, $\hat{RP}^3_\beta$, and $\hat{H}_\lambda$ for the parameter values of maximal AUC performance in dependency of the number of random walks per user. The annotations on the right-sided y-axis indicate the performance of the exact algorithms P^3_α, RP^3_β, and H_λ for the same parameter values.

Dataset	Recommender	AUC	Prec	GiniD	Pers	Surp
MovieLens-M (5 m walks)	$\hat{P}^3_\alpha$ ($\alpha = 1.8$)	0.013	0.054	2.256	0.127	0.221
	$\hat{RP}^3_\beta$ ($\beta = 0.8$)	0.089	1.188	2.813	0.148	1.190
	$\hat{H}_\lambda$ ($\lambda = 0.1$)	0.329	4.263	9.851	0.603	10.08
iPlayer (1 m walks)	$\hat{P}^3_\alpha$ ($\alpha = 1.5$)	0.014	0.138	0.050	0.011	0.008
	$\hat{RP}^3_\beta$ ($\beta = 0.7$)	0.012	0.169	0.076	0.029	0.031
	$\hat{H}_\lambda$ ($\lambda = 0.2$)	0.026	0.067	0.303	0.071	0.064
BookCrossing (1 m walks)	$\hat{P}^3_\alpha$ ($\alpha = 0.9$)	0.173	0.605	1.089	0.441	0.074
	$\hat{RP}^3_\beta$ ($\beta = 0.3$)	0.237	0.324	1.103	0.247	0.186
	$\hat{H}_\lambda$ ($\lambda = 0.6$)	0.359	0.125	1.527	0.436	0.287

Table 3: Percentage of performance deviation between P^3_α, RP^3_β, and H_λ and $\hat{P}^3_\alpha$, $\hat{RP}^3_\beta$, and $\hat{H}_\lambda$ after 1 m or 5 m random walks per user for parameter values of maximal AUC performance.

more precise than H_λ but also has higher diversity scores. For iPlayer RP^3_β is slightly more precise than WI-kNN and achieves higher diversity scores. No clear winner can be found for the MovieLens-M dataset: RP^3_β shows better precision and surprisal scores but WI-kNN succeeds in terms of GiniD and Pers performance.

6.4 Performance of Sampling Approximations

The goal of our second experiments is to investigate the performance of our sampling algorithms dependent on number of samples (i.e., number of random walks).

We determined the performance of our sampling algorithms $\hat{P}^3_\alpha$, $\hat{RP}^3_\beta$, and $\hat{H}_\lambda$ with parameter values of maximal AUC according to the non-sampling original algorithms whilst varying the number of random walks $N \in \{$ 1'000, 2'500, 5'000, 10'000, 25'000, 50'000, 100'000, 250'000, 500'000, 1 m, 2.5 m, 5 m$\}$ per user. Figure 2 shows the rate of convergence as well as the performance of the exact algorithms as indicated by the callouts near right edge of each graph. As expected the sampled algorithms' performance converge to that of the exact ones with increasing N. To illustrate the closeness of the results we computed the percentage deviation $d = (|m - \hat{m}|) * 100/m$ between the sampling procedures' $\hat{m}$ and exact calculations' m performance metrics for 5 million random walks for MovieLens-M and 1 m random walks for iPlayer and BookCrossing. The results of this procedure, listed in Table 3, show that the sampled algorithms usually deviate less than 1% from the exact ones, less than 3% in all cases but for $\hat{H}_\lambda$ for MovieLens-M. Despite the greater number of random walks, d is greater for the MovieLens-M dataset than for the iPlayer or BookCrossing datasets. We hypothesize that this is due to the greater number of distinct paths of length three starting at a given user existing in the graph G for MovieLens-M dataset as indicated by the high average vertex degree of 71.8 (compared to iPlayer: 7.1, BookCrossing: 11.6).

Furthermore, Figure 2 clearly indicates that $\hat{P}^3_\alpha$ requires less samples to converge than $\hat{RP}^3_\beta$, which in turn converges faster than $\hat{H}_\lambda$. Since these algorithms can be computed using Algorithm 1 and differ only in c_{rw}, we hypothesize that c_{rw} controls the efficiency of sampling. As a result $\hat{P}^3_\alpha$ is the most accurate sampling algorithm for small values of N. For slightly greater N, $\hat{RP}^3_\beta$ is more accurate than $\hat{P}^3_\alpha$ in the MovieLens-M and iPlayer datasets. If we increase N even further, $\hat{H}_\lambda$ becomes the most accurate recommender for the iPlayer and BookCrossing dataset.

Considering recommendation accuracy, diversity, and the sample size required to obtain acceptable accuracy, our re-

sults suggest the following: On data with moderate sparsity and balanced user and item degrees (MovieLens-M) one should use $\hat{\mathrm{P}}^3_\alpha$ if computing resources are scarce, i.e., $N < 250'000$, because of the algorithm's better precision and otherwise $\mathrm{R}\hat{\mathrm{P}}^3_\beta$ which provides best accuracy and diversity (at comparable level of accuracy). For sparser data with more ratings per item than per user on average (iPlayer), $\mathrm{R}\hat{\mathrm{P}}^3_\beta$ is probably the best choice since it reaches almost the maximal accuracy but gives better diversity (at comparable level of accuracy) and converges quicker than $\hat{\mathrm{H}}_\lambda$. For a sparse dataset with an average item degree smaller than the average user degree (BookCrossing) $\hat{\mathrm{H}}_\lambda$ is the best choice given that computing resources are plenty ($N > 25'000$), since it gives better precision and diversity (at comparable level of accuracy). In the case of limited computing power however, the choice is not obvious due to the poor accuracy of $\mathrm{R}\hat{\mathrm{P}}^3_\beta$ and $\hat{\mathrm{H}}_\lambda$ and very poor diversity of $\hat{\mathrm{P}}^3_\alpha$.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we studied accuracy and diversity of vertex ranking algorithms using random walk sampling techniques and thereby bring together three streams of earlier presented work. Specifically, we introduced RP^3_β, a novel graph random walk based recommendation algorithm based on a re-ranking of P^3 that gives better recommendation accuracy and diversity than previously proposed vertex ranking algorithms. We showed that re-ranking improves the accuracy performance over P^3 and its parameterized version P^3_α and pushes "wallflowers", i.e., long-tail items, closer to the top of the recommendation list. Our method is also competitive with another graph-based recommender H_λ that optimizes the accuracy diversity trade-off. We also showed that RP^3_β is competitive with traditional algorithms.

Additionally, we presented scalable random walk sampling implementations of the three best vertex ranking algorithms. We showed empirically that these algorithms converge to their exact counterparts with increasing number of samples. The sampling procedures have the favorable property of being anytime algorithms: a recommendation list of low accuracy can be generated after a short processing time, while longer computations, i.e., gathering more random walk samples, improve the accuracy of the recommendation list.

In future work we hope to investigate the sensitivity of the convergence of the sampling algorithms to domain characteristics and further explore convergence behavior for different datasets and algorithms. Also, we would like to take detailed run-time measurements to ascertain wall-clock time advantages and trade-offs.

Our results indicate that the goal of scalable, accurate, and surprising recommendations could be achieved with vertex ranking algorithms using random walk sampling.

Acknowledgements: We would like to thank the Hasler Foundation for their generous support under grant # 11072.

8. REFERENCES

[1] G. Adomavicius and Y. Kwon. Maximizing Aggregate Recommendation Diversity: A Graph-Theoretic Approach. In *Workshop on Novelty and Diversity in Recommender Systems, ACM RecSys*, 2011.

[2] G. Adomavicius and Y. Kwon. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[3] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting Hatches an Egg: A New Graph-Theoretich Approach to Collaborative Filtering. In *ACM SIGKDD*, 1999.

[4] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video Suggestion and Discovery for YouTube: Taking Random Walks Through the View Graph. In *WWW Conference*, 2008.

[5] T. Bogers. Movie Recommendation using Random Walks over the Contextual Graph. In *Workshop on Context-Aware Recommender Systems, ACM RecSys*, 2010.

[6] C. Cooper, S. H. Lee, T. Radzik, and Y. Siantos. Random Walks in Recommender Systems: Exact Computation and Simulations. In *WWW Conference*, 2014.

[7] P. Cremonesi, F. Garzotto, S. Negro, A. V. Papadopoulos, and R. Turrin. Looking for "Good" Recommendations: A Comparative Evaluation of Recommender Systems. In *INTERACT 2011*. Springer, 2011.

[8] D. M. Fleder and K. Hosanagar. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Management science*, 2009.

[9] F. Fouss, A. Pirotte, and M. Saerens. A Novel Way of Computing Similarities between Nodes of a Graph, with Application to Collaborative Recommendation. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.

[10] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A Free Recommender System Library. In *ACM RecSys*, 2011.

[11] D. G. Goldstein and D. C. Goldstein. Profiting from the long tail. *Harvard Business Review*, 2006.

[12] M. Gori and A. Pucci. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *IJCAI*, 2007.

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 2004.

[14] Z. Huang, H. Chen, and D. Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. *ACM Transactions on Information Systems*, 2004.

[15] M. Jamali and M. Ester. TrustWalker: A RandomWalk Model for Combining Trust-based and Item-based Recommendation. In *ACM SIGKDD*, 2009.

[16] S. Lee, S. Park, M. Kahng, and S.-g. Lee. PathRank: A Novel Node Ranking Measure on a Heterogeneous Graph for Recommender Systems. In *CIKM*, 2012.

[17] S. M. McNee, J. Riedl, and J. A. Konstan. Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems. In *CHI'06 extended abstracts*. ACM, 2006.

[18] E. Pariser. *The Filter Bubble: What the Internet is Hiding from You*. Penguin UK, 2011.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, 2009.

[20] P. Sarkar, A. W. Moore, and A. Prakash. Fast Incremental Proximity Search in Large Graphs. In *ACM ICML*, 2008.

[21] S. Vargas and P. Castells. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *ACM RecSys*, 2011.

[22] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal Recommendation on Graphs via Long- and Short-term Preference Fusion. In *ACM SIGKDD*, 2010.

[23] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *PNAS*, 2010.

[24] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving Recommendation Lists Through Topic Diversification. In *WWW Conference*, 2005.